Trading Cores for Memory Bandwidth in Real-time Systems

Ahmed Alhammad and Rodolfo Pellizzoni
Outline

• Introduction
• Problem Description
• Our Solution
• Conclusion
Parallel Tasks

DAG Model
Parallel Tasks

- Two notable scheduling schemes:
  1. Global
  2. Federated

DAG Model
Federated Scheduling

• High utilization tasks \( (u_i \geq 1) \) are assigned dedicated cores.
• Low utilization tasks \( (u_i < 1) \) are scheduled as multiprocessor scheduling of sequential tasks.
Federated Scheduling

• **High utilization tasks** \((u_i \geq 1)\) are assigned dedicated cores.
• Low utilization tasks \((u_i < 1)\) are scheduled as multiprocessor scheduling of sequential tasks.
Outline

• Introduction
• **Problem Description**
• Our Solution
• Conclusion
The Problem

core$_1$ core$_2$ core$_3$ core$_4$ core$_5$

off-chip memory

task$_1$

task$_2$

task$_3$
The Problem

- core_1
- core_2
- core_3
- core_4
- core_5
- off-chip memory
- task_1
- task_2
- task_3
The Problem

core_1  core_2  core_3  core_4  core_5

off-chip memory

task_1  task_2  task_3
The Problem

- core\textsubscript{1}
- core\textsubscript{2}
- core\textsubscript{3}
- core\textsubscript{4}
- core\textsubscript{5}

- off-chip memory

- task\textsubscript{1}
- task\textsubscript{2}
- task\textsubscript{3}
The Problem

- Tasks interfere through shared main memory
Outline

• Introduction
• Problem Description
• **Our Solution**
• Conclusion
Outline

• Introduction
• Problem Description
• **Our Solution**
  1. Makespan Bound
  2. Round-robin Arbitration
  3. Trading Cores for Memory Bandwidth
  4. Three Algorithms
  5. Results
• Conclusion
Makespan Bound (greedy scheduling)

$m_i$ number of cores
$q_i$ bandwidth fraction
Makespan Bound (greedy scheduling)

• Each DAG task is characterized by:
  1. Computation volume \( C_i^e \)
  2. Memory volume \( C_i^m \)
  3. Computation critical path \( L_i \)

\( m_i \) number of cores
\( q_i \) bandwidth fraction
Makespan Bound (greedy scheduling)

• Each DAG task is characterized by:
  1. Computation volume $C_i^e$
  2. Memory volume $C_i^m$
  3. Computation critical path $L_i$

\[ C_i = \frac{C_i^m}{q_i} + \frac{C_i^e - L_i}{m_i} + L_i \leq D_i \]

$m_i$ number of cores
$q_i$ bandwidth fraction
Makespan Bound (greedy scheduling)

- Each DAG task is characterized by:
  1. Computation volume $C_i^e$
  2. Memory volume $C_i^m$
  3. Computation critical path $L_i$

$$C_i = \frac{C_i^m}{q_i} + \frac{C_i^e - L_i}{m_i} + L_i \leq D_i$$

$m_i$ number of cores
$q_i$ bandwidth fraction
Outline

• Introduction
• Problem Description
• **Our Solution**
• Conclusion

1. Makespan Bound
2. **Round-robin Arbitration**
3. Trading Cores for Memory Bandwidth
4. Three Algorithms
5. Results
Round-robin Arbitration

request buffer

task_i task_j
Round-robin Arbitration

- $C_i^m + C_i^m \times (\sum_{j \neq i} m_j)$

request buffer

task$_i$  

request buffer

task$_j$
Round-robin Arbitration

- $C_i^m + C_i^m \times (\sum_{j \neq i} m_j)$
Round-robin Arbitration

- \( C_i^m + C_i^m \times (\sum_{j \neq i} m_j) \)

- \( C_i^m + C_i^m \times (n - 1) \)
Outline

• Introduction
• Problem Description
• Our Solution
  1. Makespan Bound
  2. Round-robin Arbitration
  3. Trading Cores for Memory Bandwidth
  4. Three Algorithms
  5. Results
• Conclusion
Trading Cores for Memory Bandwidth

• Tasks have different demand regarding memory
• The number of cores is expected to increase
• Memory bandwidth is also growing but in slower rate
Trading Cores for Memory Bandwidth

• Tasks have different demand regarding memory
• The number of cores is expected to increase
• Memory bandwidth is also growing but in slower rate
• The idea of trading cores for memory bandwidth
Example
Example

• Consider a task with the following parameters:
  \( C^m_1 = 50, \ C^e_1 = 100 \) and \( D_1 = 150 \)
Example

• Consider a task with the following parameters: $C_1^m = 50$, $C_1^e = 100$ and $D_1 = 150$

• For simplicity, assume $L_1 = 0$
Example

• Consider a task with the following parameters:
  \( C_1^m = 50, C_1^e = 100 \) and \( D_1 = 150 \)
• For simplicity, assume \( L_1 = 0 \)
• We recall that the makespan bound is:

  \[ C_i(q_i,m_i) = \frac{C_i^m}{q_i} + \frac{C_i^e - L_i}{m_i} + L_i \]
Example

• Consider a task with the following parameters:
  \( C_1^m = 50, \ C_1^e = 100 \) and \( D_1 = 150 \)
• For simplicity, assume \( L_1 = 0 \).
• We recall that the makespan bound is:

\[
C_i(q_i,m_i) = \frac{c_i^m}{q_i} + \frac{c_i^e - L_i}{m_i} + L_i
\]

• The makespan \( C_1(1,1) = 50 \times 1 + 100/1 = 150 \)
Example

• Consider a task with the following parameters:
  \[ C_1^m = 50, \quad C_1^e = 100 \quad \text{and} \quad D_1 = 150 \]
• For simplicity, assume \( L_1 = 0 \).
• We recall that the makespan bound is:

\[
C_i(q_i, m_i) = \frac{C_i^m}{q_i} + \frac{C_i^e - L_i}{m_i} + L_i
\]

• The makespan \( C_1(1,1) = 50 \times 1 + 100/1 = 150 \)
• The makespan \( C_1(1/2, 2) = 50 \times 2 + 100/2 = 150 \)
Outline

• Introduction
• Problem Description
• **Our Solution**
• Conclusion

1. Makespan Bound
2. Round-robin Arbitration
3. Trading Cores for Memory Bandwidth
4. Three Algorithms
5. Results
Assignment Algorithm

• Assign each task $q_i$ and $m_i$ such that

$$\sum_{i=1}^{n} q_i \leq 1 \text{ and } \sum_{i=1}^{n} m_i \leq m$$

• We propose three algorithms:
  (1) Optimal algorithm with $q_i \in \mathbb{R}$
  (2) Harmonic RR with $q_i = 1/2^j$
  (3) Software-based regulation
Optimal-Assign Algorithm \( (q_i \in \mathbb{R}) \)

- \[ C_i = \frac{c_i^m}{q_i} + \frac{c_i^e - L_i}{m_i} + L_i \leq D_i \]
Optimal-Assign Algorithm \((q_i \in \mathbb{R})\)

- \[ C_i = \frac{c_i^m}{q_i} + \frac{c_i^e - L_i}{m_i} + L_i \leq D_i \]
- \[ q_i(m_i) = \frac{c_i^m \times m_i}{(D_i - L_i) \times m_i - (c_i^e - L_i)} \]
Optimal-Assign Algorithm \((q_i \in \mathbb{R})\)

- \(C_i = \frac{c_i^m}{q_i} + \frac{c_i^e - L_i}{m_i} + L_i \leq D_i\)
- \(q_i(m_i) = \frac{c_i^m \times m_i}{(D_i - L_i) \times m_i - (c_i^e - L_i)}\)
Optimal-Assign Algorithm ($q_i \in \mathbb{R}$)

- $C_i = \frac{c_i^m}{q_i} + \frac{c_i^e-L_i}{m_i} + L_i \leq D_i$

- $q_i(m_i) = \frac{c_i^m \times m_i}{(D_i-L_i) \times m_i-(c_i^e-L_i)}$

- $\Delta_i(m_i) = q_i(m_i) - q_i(m_i + 1)$
Optimal-Assign Algorithm ($q_i \in \mathbb{R}$)

- $C_i = \frac{c_i^m}{q_i} + \frac{c_i^e - L_i}{m_i} + L_i \leq D_i$
- $q_i(m_i) = \frac{c_i^m \times m_i}{(D_i - L_i) \times m_i - (c_i^e - L_i)}$
- $\Delta_i(m_i) = q_i(m_i) - q_i(m_i + 1)$
Optimal-Assign Algorithm ($q_i \in \mathbb{R}$)

- $C_i = \frac{c_i^m}{q_i} + \frac{c_i^e - L_i}{m_i} + L_i \leq D_i$

- $q_i(m_i) = \frac{c_i^m \times m_i}{(D_i - L_i) \times m_i - (c_i^e - L_i)}$

- $\Delta_i(m_i) = q_i(m_i) - q_i(m_i + 1)$

- $\Delta_i(m_i) > 0$ and $\Delta_i(m_i) > \Delta_i(m_i + 1)$

The function $\Delta_i(m_i)$ is decreasing and convex.
Harmonic RR \( (q_i \in \{1/2^j\}) \)
Harmonic RR \((q_i \in \{1/2^j\})\)

- \(\Theta_i(q_i) = \frac{q_i - \frac{q_i}{2}}{m_i \left(\frac{q_i}{2}\right) - m_i(q_i)}\)
Harmonic RR \( (q_i \in \{1/2^j\}) \)

\[ \Theta_i(q_i) = \frac{q_i - \frac{q_i}{2}}{m_i(\frac{q_i}{2}) - m_i(q_i)} \]

• We design the algorithm to continue until achieving 100% bandwidth utilization.
Example

\[ q_1 + q_2 = 2 \]
Example

\[ q_1 + q_2 = 1.5 \]
Example

\[ q_1 + q_2 = 1.25 \]
Example

\[ q_1 + q_2 = 0.75 \]
Example

\[ q_1 + q_2 = 1.5 \]
Example

\[ q_1 + q_2 = 1 \]
Memguard
Memguard

- Each task is assigned a budget $Q_i$
Memguard

• Each task is assigned a budget $Q_i$
• When a task finishes this budget, it is suspended
Memguard

• Each task is assigned a budget $Q_i$
• When a task finishes this budget, it is suspended
• The budget is recharged every period $P$
Memguard

• Each task is assigned a budget $Q_i$
• When a task finishes this budget, it is suspended
• The budget is recharged every period $P$
How to Compute Memory Time?
How to Compute Memory Time?

• Assume the following task:

memory

computation
How to Compute Memory Time?

• Assume the following task:

\[ Q_i - \epsilon \quad Q_i \quad \text{suspension} \quad \epsilon \]
How to Compute Memory Time?

• Assume the following task:
How to Compute Memory Time?

• Assume the following task:
How to Compute Memory Time?

• Assume the following task:

\[ Q_i - \epsilon \]

memory computation

\[ Q_i \]

suspension

\[ \epsilon \]

\[ Q_i \]

suspension

\[ Q_i \]

suspension

\[ P - Q_i \]

\[ P - Q_i \]

\[ \Omega_i \]

contention

\[ \Omega_i \]

contention

\[ \Omega_i \]

contention
How to Compute Memory Time?

• Assume the following task:

\[ Q_i - \epsilon \quad Q_i \quad \text{suspension} \quad \epsilon \]

\[ Q_i \quad \text{suspension} \quad Q_i \quad \text{suspension} \]

\[ \Omega_i \quad \text{contention} \quad \Omega_i \quad \text{contention} \quad \Omega_i \quad \text{contention} \]

\[ P - Q_i \quad P - Q_i \]
Memory Time Under Regulation

\[ \left| \frac{c_i^m}{Q_i} \right| \times P + P \]
Memory Time Under Regulation

- $\left\lfloor \frac{c_i^m}{Q_i} \right\rfloor \times P + P$

- We let $\frac{P}{Q_i} = \frac{1}{q_i}$ and remove the floor function
Memory Time Under Regulation

- \( \left\lfloor \frac{c_i^m}{Q_i} \right\rfloor \times P + P \)

- We let \( \frac{P}{Q_i} = \frac{1}{q_i} \) and remove the floor function

- \( \frac{c_i^m}{q_i} + P \) which is the same as in Optimal-Assign but shifted by constant \( P \)
Memory Time Under Regulation

- \[ \left\lfloor \frac{c^m_i}{Q_i} \right \rfloor \times P + P \]
- We let \( \frac{P}{Q_i} = \frac{1}{q_i} \) and remove the floor function
- \( \frac{c^m_i}{q_i} + P \) which is the same as in Optimal-Assign but shifted by constant \( P \)
Memory Time Under Regulation

- \[ \left\lfloor \frac{c_i^m}{Q_i} \right\rfloor \times P + P \]
- We let \( P = \frac{1}{Q_i} = \frac{1}{q_i} \) and remove the floor function
- \( \frac{c_i^m}{q_i} + P \) which is the same as in Optimal-Assign but shifted by constant \( P \)
Outline

• Introduction
• Problem Description

• **Our Solution**

• Conclusion

1. Makespan Bound
2. Round-robin Arbitration
3. Trading Cores for Memory Bandwidth
4. Three Algorithms

5. Results
\[ m = 32, n = 5 \text{ and } U^e = 10 \]
$m = 32, n = 5$ and $U^e = 10$
\[ m = 32, \ n = 5 \text{ and } U^e = 10 \]
\( m = 32, n = 5 \) and \( U^e = 10 \)
\[ m = 32, n = 5 \text{ and } U^e = 10 \]
\( m = \infty, n = 5 \) and \( U^e = 10 \)
Outline

• Introduction
• Problem Description
• Our Solution  → 1. Makespan Bound
• Conclusion  → 2. Round-robin Arbitration
              → 3. Trading Cores for Memory Bandwidth
              → 4. Three Algorithms
              → 5. Results
Conclusion

• Federated scheduling is an elegant approach for parallel tasks
Conclusion

• Federated scheduling is an elegant approach for parallel tasks
• Previous research did not consider memory time
Conclusion

• Federated scheduling is an elegant approach for parallel tasks
• Previous research did not consider memory time
• In this work, we show how to integrate memory time
Conclusion

• Federated scheduling is an elegant approach for parallel tasks
• Previous research did not consider memory time
• In this work, we show how to integrate memory time
• We introduce a novel method to trade cores for memory bandwidth
Conclusion

• Federated scheduling is an elegant approach for parallel tasks
• Previous research did not consider memory time
• In this work, we show how to integrate memory time
• We introduce a novel method to trade cores for memory bandwidth
• We propose three algorithms for this method
Conclusion

• Federated scheduling is an elegant approach for parallel tasks
• Previous research did not consider memory time
• In this work, we show how to integrate memory time
• We introduce a novel method to trade cores for memory bandwidth
• We propose three algorithms for this method
• The results indicate a significant improvement over nRR